

A standard transformation from XML to RDF via XSLT

F. Breitling*

Astrophysikalisches Institut Potsdam, An der Sternwarte 16, D-14482 Potsdam, Germany

Received Nov 2008, accepted Jun 2009

Published online later

Key words standards, semantic astronomy

A generic transformation of XML data into the Resource Description Framework (RDF) and its implementation by XSLT transformations is presented. It was developed by the grid integration project for robotic telescopes of AstroGrid-D to provide network communication through the Remote Telescope Markup Language (RTML) to its RDF based information service. The transformation's generality is explained by this example. It automates the transformation of XML data into RDF and thus solves this problem of semantic computing. Its design also permits the inverse transformation but this is not yet implemented.

© 2009 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim

1 Introduction

Progress in astronomy has required experiments of ever increasing complexity. Modern experiments have high computational demands and produce tremendous amounts of data. In reaction to these demands new technologies such as grid and semantic computing have been applied to astronomy leading to the fields of astronomical grid computing and semantic astronomy. For example the grid project of the German astronomy community (AstroGrid-D) (Enke et al. 2007), the *German Astrophysical Virtual Observatory* or the *International Virtual Observatory Alliance* have been active in these fields. Recent efforts in semantic astronomy have focused on the usage of the Resource Description Framework¹ (RDF) for metadata management. This approach was also pursued in AstroGrid-D with its information service *Stellaris* (Högqvist, Röblitz & Reinefeld 2007). The motivation is to create a *semantic web*, i.e. a collection of information with a simple, more machine friendly structure. RDF is a new data model which describes data through graphs of *resources*, which are accessible through Uniform Resource Identifiers (URIs). Complex resources are hierarchically composed of simpler ones in analogy to the real-world object they describe. For example a telescope is composed of a camera, which has a filter wheel, which has filters, etc. This concept makes RDF an interesting choice for the metadata management in heterogeneous software environments, where automatic interaction between different components is desired. For example a resource broker in a telescope network needs to find telescopes with a certain filter.

However, a general problem arises when RDF is to be applied, since traditional data formats are not RDF com-

plaint. In astronomy for example information is often stored in various XML dialects. Individual solutions for mapping a particular dialect into RDF require a lot of effort and thus are often not feasible. This constitutes a considerable barrier for the application of RDF to this rich heritage of XML data. What is needed is a generic transformation to map any XML dialect into RDF. This problem has been addressed before, e.g. in the work by Trastour, Preist & Coleman (2004), Battle (2006) and Akhtar et al. (2008). However, their solutions are complex and require software which is not yet available or does not work for the example discussed here. Another approach is offered through the *Virtuoso* database². It provides means to store XML data in relational databases and to dynamically convert it into RDF. However, this conversion also requires a data specific schema and therefore can not immediately be applied. Here a generic transformation for the automatic conversion of XML into RDF is presented, which solves this problem for any XML dialect.

To illustrate the problem and its solution the application to OpenTel³ (Breitling, Granzer & Enke 2008, Granzer et al. 2007), the grid integration project for robotic telescopes of AstroGrid-D, is discussed. The goal of this project is the integration of heterogeneous robotic telescopes into a common infrastructure with standard components to build a global telescope network. The infrastructure is provided by AstroGrid-D. The metadata management is facilitated by the RDF information service *Stellaris*. Its purpose in a robotic telescope network is to provide information e.g. about telescopes, weather and scheduled observations. In OpenTel this information is exchanged in the Remote Telescope Markup Language (RTML, Hessman 2006). RTML is an XML dialect developed by the *Heterogeneous Telescope Network* (Allan et al. 2006). A transformation to

* e-mail: fbreitling@aip.de

¹ <http://www.w3.org/RDF/>

² <http://www.openlinksw.com/virtuoso/>

³ <http://www.gac-grid.org/project-products/RoboticTelescopes.html>

RDF is needed to make the RTML data accessible to Stellaris. Here a general transformation appears useful, since also other XML data such as Usage Records⁴ of grid jobs is provided to Stellaris e.g. for monitoring.

2 Design

A transformation to RDF has to create the URIs of its resources and connect them through the RDF *triple* structure consisting of subject, predicate and object. However, this can be done in different ways which leaves room for additional design criteria. The following have been added here:

- avoidance of blank nodes,
- one-to-one mapping for a bidirectional extension,
- independence of XML schema.

Blank nodes are subjects without name, i.e. URI. Therefore direct access to them is not possible and some operations in RDF databases such as replacement of nodes cannot be performed. Other disadvantages include reduced performance of SPARQL processors and RDF databases. By giving every node a URI these problems are avoided and direct access is possible.

One-to-one mapping means, that the transformation is injective and therefore the original XML can uniquely be reconstructed from RDF through the inverse transformation. A bidirectional transformation can be important e.g. in a robotic telescope network where information about scheduled observations is stored in RDF but where rescheduling requires the original RTML observation request. Therefore in AstroGrid-D also the RTML observation requests were stored along with the RDF. The inverse transformation could make this additional service unnecessary. A unique reconstruction of the original XML requires e.g. to preserve the distinction between attributes and elements. As shown below, this is accomplished by the different transformation of attributes and elements.

Independence from the XML schema means, that the transformation does not require information contained in the XML schema. Therefore it can always directly be applied and produce the same result. An example for information only contained in the schema are XML sequences. XML sequences fix the order in which elements have to appear. Therefore the order of elements might be important for some parts of the data and thus should be preserved. It is then also important for a later inverse transformation. A schema independent transformation can only preserve or ignore the order information of all elements. This transformation preserves this information by default, but it can be disabled if it is not relevant for a particular data.

3 Transformation

In the following sections the transformation is explained for the simplified RTML description of the robotic telescope STELLA-I (Strassmeier et al. 2004) in listing 1.

⁴ <http://staff.psc.edu/lfm/PSC/Grid/UR-WG/UR-Spec-gfd.58.pdf>

RDF has different representations such as RDF/XML, N-Triples, N3 and Turtle. For the following discussion the Turtle notation is used because of its compact and clear structure. It represents triples in the order of subject, predicate and object. Predicates and objects of the same subject can be grouped in a block. Resources are enclosed by angle brackets, literals by quotation marks. An example of the Turtle syntax is shown in listing 2. It is the RDF counterpart to the RTML description generated by the presented transformation.

Since the target is the transformation of XML dialects, XSLT transformations⁵ are used. XSLT transformations were specifically developed to facilitate the conversion from one XML format to another. They are programmed through an XSLT stylesheet. The stylesheet discussed here is name `xml2rdf3.xsl` and freely available through the project page⁶. Its output is RDF/XML. The corresponding Turtle representation in listing 2 has been produced with the RDF parser *Raptor*.

The transformation generates no base URI by default. Thus all resources have relative URIs. Relative URIs are useful where a completion of the base URI is done by an RDF service such as Stellaris. The RDF is then dereferencable for RDF services of arbitrary URI. In contrast, RDF with absolute URIs is only dereferencable for one particular service URI and might require a later replacement. However, if desired a base URI can be added via the `BaseURI` variable in the XSLT stylesheet.

The following sections explains the transformation of the different XML components. A graphical representation is shown in Fig. 1 to support the discussion. It was obtained with the RDF graph visualization tool *RDF Gravity*⁷.

3.1 Transformation of Attributes

XML attributes have a natural correspondence to triples since they contain two pieces of information. This suggests their transformation into predicate and object of the parent element which itself serves as subject. For example the `units` attribute of the `FocalLength` element at line 7 of listing 1 transforms into the triple at line 23-24 of listing 2.

3.2 Transformation of Text

A natural representation of XML text is provided by RDF literals. Literals can be further specified by type and language information, but this is not used here. Literals can only serve as objects. The subject is already provided by the parent element as defined by the attribute transformation above. The predicate can be provided by the RDF utility class `rdf:value`. Again the `FocalLength` element at line 7 of listing 1 serves as an example. Its text 9 . 6 transforms into the triple at line 23 and 25 of listing 2.

⁵ <http://www.w3.org/TR/xslt/>

⁶ <http://www.gac-grid.org/project-products/Software/XML2RDF.html>

⁷ <http://semweb.salzburgresearch.at/apps/rdf-gravity/>

Listing 1: Description of robotic telescope STELLA-I.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <RTML version="3.2" mode="resource" uid="rtml
  ://www.opentel.net/STELLA-I"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
4   xmlns="http://www.rtml.org/v3.2" xsi:
    schemaLocation="RTML-v3.2.xsd">
5   <!-- This is only a fragment -->
6   <Telescope>
7     <FocalLength units="meters">9.6</
      FocalLength>
8     <Camera>
9       <FilterWheel>
10      <Filter type="Johnson_U" name="U"/>
11      <Filter type="Johnson_B" name="B"/>
12    </FilterWheel>
13  </Camera>
14 </Telescope>
15 </RTML>

```

3.3 Transformation of Elements

Child elements cannot be modeled with literal objects as before but require resources. The necessary URIs are constructed from the sequence of ancestor elements. In case of child elements with the same name, an ID is also appended, to make the URIs unique. The parent element provides the predicate. The *FilterWheel* and its two *Filter* elements line 9-12 of listing 1 serve as example. The filter elements result in two resource at line 35-41 of listing 2. A 2 is appended as unique ID to the second URI separated by an underscore.

In addition to the predicate from the parent element (line 30-31), another predicate (*rdf:_no*) is added to conserve the element's order (line 32-33). However this is only added where necessary, i.e. to elements with siblings. It can be disabled (commented out) in the XSLT stylesheet if it is not relevant for the XML data.

3.4 Transformation of Comments

XML comments are preserved in additional triples to the element where they occur. *xs:comment* serves as predicate. An example is given at line 5 of listing 1 and the corresponding lines 6 and 12 of listing 2.

3.5 Execution

The application of the XSLT stylesheet requires an XSLT version 1.0 compliant processor such as *xsltproc*. The command line syntax is:

```
xsltproc xml2rdf3.xsl STELLA-I.rtml
```

4 Conclusion

A generic transformation for XML data into RDF/XML has been presented. Owing to its generality it can be applied to any XML dialect and serve as standard to automate the

Listing 2: RDF (Turtle) representation of listing 1.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-
  syntax-ns#> .
2 @prefix ns1: <http://www.rtml.org/v3.2#> .
3 @prefix xsi: <http://www.w3.org/2001/XMLSchema-
  instance#> .
4 @prefix xs: <http://www.w3.org/TR/2008/REC-xml-
  20081126#> .
5
6 <file:///STELLA-I_3.rdf#RTML>
7   ns1:Telescope <file:///STELLA-I_3.rdf#RTML/
      Telescope> ;
8   ns1:mode "resource" ;
9   ns1:uid "rtml://www.opentel.net/STELLA-I" ;
10  ns1:version "3.2" ;
11  xsi:schemaLocation "RTML-v3.2.xsd" ;
12  xs:comment " This is only a fragment " .
13
14 <file:///STELLA-I_3.rdf>
15   ns1:RTML <file:///STELLA-I_3.rdf#RTML> .
16
17 <file:///STELLA-I_3.rdf#RTML/Telescope>
18   ns1:Camera <file:///STELLA-I_3.rdf#RTML/
      Telescope/Camera> ;
19   ns1:FocalLength <file:///STELLA-I_3.rdf#
      RTML/Telescope/FocalLength> ;
20   rdf:_1 <file:///STELLA-I_3.rdf#RTML/
      Telescope/FocalLength> ;
21   rdf:_2 <file:///STELLA-I_3.rdf#RTML/
      Telescope/Camera> .
22
23 <file:///STELLA-I_3.rdf#RTML/Telescope/
  FocalLength>
24   ns1:units "meters" ;
25   rdf:value "9.6" .
26
27 <file:///STELLA-I_3.rdf#RTML/Telescope/Camera>
28   ns1:FilterWheel <file:///STELLA-I_3.rdf#
      RTML/Telescope/Camera/FilterWheel> .
29
30 <file:///STELLA-I_3.rdf#RTML/Telescope/Camera/
  FilterWheel>
31   ns1:Filter <file:///STELLA-I_3.rdf#RTML/
      Telescope/Camera/FilterWheel/Filter> , <
      file:///STELLA-I_3.rdf#RTML/Telescope/
      Camera/FilterWheel/Filter_2> ;
32   rdf:_1 <file:///STELLA-I_3.rdf#RTML/
      Telescope/Camera/FilterWheel/Filter> ;
33   rdf:_2 <file:///STELLA-I_3.rdf#RTML/
      Telescope/Camera/FilterWheel/Filter_2>
34   .
35 <file:///STELLA-I_3.rdf#RTML/Telescope/Camera/
  FilterWheel/Filter>
36   ns1:name "U" ;
37   ns1:type "Johnson_U" .
38
39 <file:///STELLA-I_3.rdf#RTML/Telescope/Camera/
  FilterWheel/Filter_2>
40   ns1:name "B" ;
41   ns1:type "Johnson_B" .

```

transformation of XML data into RDF. Thus it solves this conversion problem of semantic computing. Its application is simple and has proven useful to semantic astronomy and grid computing. The transformation is injective so that the inverse transformation as bidirectional extension can be developed.

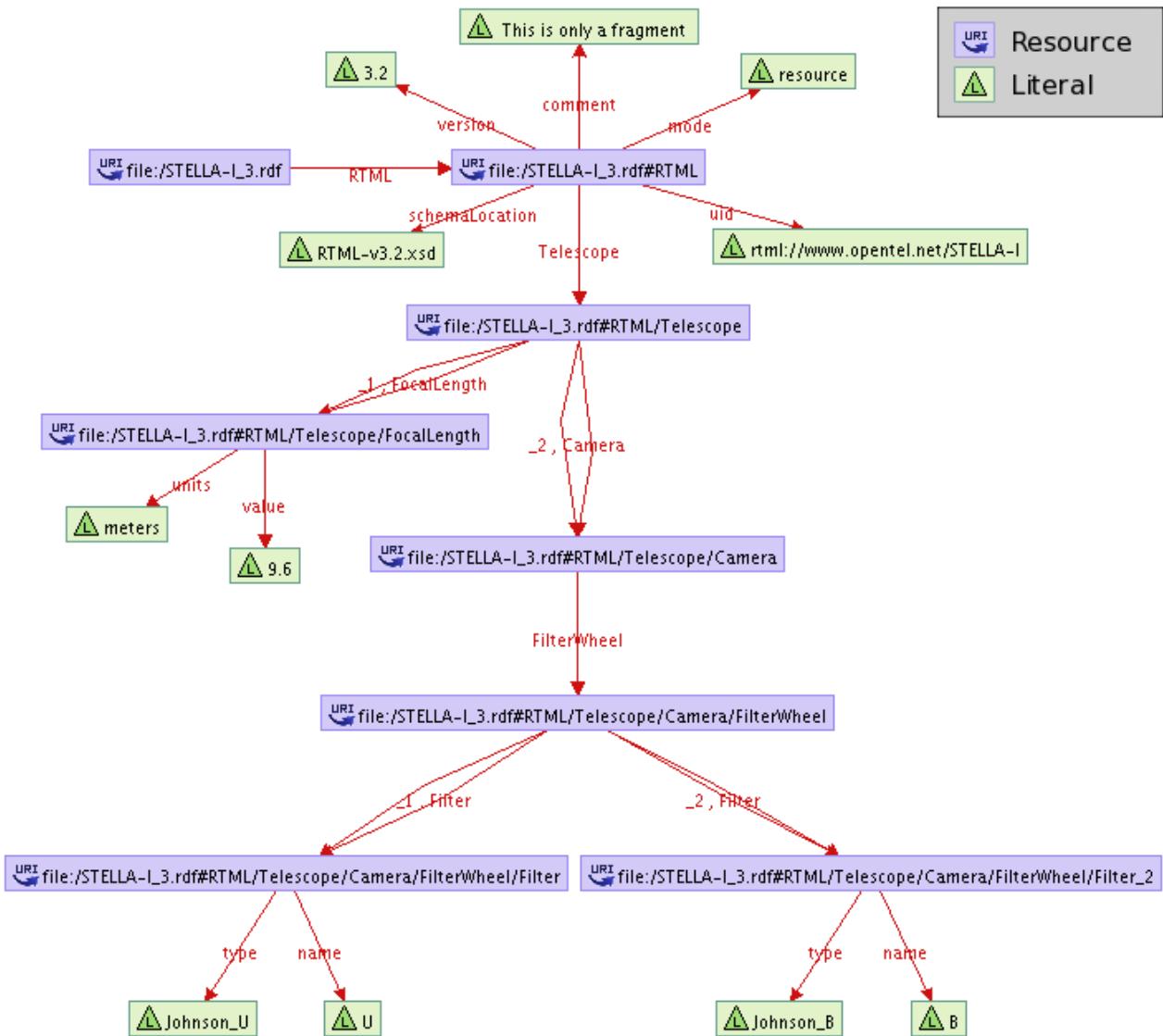


Fig. 1: Graphical representation of the RDF graph of listing 2 describing parts of the robotic telescope STELLA-I. Resources and literals are represented by purple and green labels, respectively (see legend top left). Labeled arrows represent predicates which connect subjects with objects to triples. The graphic was produced with *RDF Gravity*. This RDF visualization tool also filled in the file name as base URI. The missing two backslashes after the colon is an unconventional feature of this tool.

Acknowledgements. I would like to thank my colleagues Mikael Höggqvist, Harry Enke and Stevan White for helpful discussion and useful comments. Moreover many thanks goes to Harry Enke, Matthias Steinmetz and the German *Federal Ministry of Education and Research* (BMBF) for supporting this work within AstroGrid-D and the D-Grid initiative.

References

Akhtar, W., Kopecky, J., Krennwallner, T., Polleres, A.: 2008 in: S. Bechhofer, M. Hauswirth, J. Hoffmann, M. Koubarakis (eds.), *The Semantic Web: Research and Applications*, LNCS 5021, 432

Allan, A., Hessman, F. V., Bishoff, K. et al.: 2006, AN, 327, 744

Battle, S.: 2006 in: I. Dickinson, *First Jena User Conference*, Bristol, UK

Breitling, F., Granzer, T., Enke, H.: 2008, AN 329, 342

Enke, H., Steinmetz, M., Radke, T., Reiser, A., Röblitz, T., Höggqvist, M.: 2007 in: R. Maschuw et al. (eds.), *German e-Science Conference*, ID 316645.0

Granzer, T., Breitling, F., Braun, M., Enke, H., Röblitz, T.: 2007 in: R. Maschuw et al. (eds.), *German e-Science Conference*, ID 316644.0

Hessman, F. V.: 2006, AN 327, 751

Höggqvist, M., Röblitz, T., Reinefeld, A.: 2007 in: R. Maschuw et al. (eds.), *German e-Science Conference*, ID 316518.0

Strassmeier, K. G., Granzer, T., Weber, M. et al.: 2004, AN 325, 527

Trastour, D., Ferdinand, M., Zirpins, C.: 2004 in: *EDOC 2003, IEEE Computer Society 2003*, 222

Listing 3: XSLT stylesheet for the transformation from XML to RDF as found at the project page at <http://www.gac-grid.org/project-products/Software/XML2RDF.html>. It is not part of the publication in Astronomical Notes. Only the reference to the project page is included.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml2rdf3.xsl          XSLT stylesheet to transform XML into RDF/XML
3
4     Version          3.0 (2009-05-28)
5     Changes to V2.5  rdf:value for all text, no attribute triples,
6                      order predicates, comments as triples
7     Web page         http://www.gac-grid.org/project-products/Software/XML2RDF.html
8     Usage            xsltproc xml2rdf3.xsl file.xml
9     Author           Frank Breitling (fbreitling at aip.de)
10    Copyright 2009  AstroGrid-D (http://www.gac-grid.org/)
11
12    Licensed under the Apache License, Version 2.0 (the "License");
13    you may not use this file except in compliance with the License.
14    You may obtain a copy of the License at
15
16        http://www.apache.org/licenses/LICENSE-2.0
17
18    Unless required by applicable law or agreed to in writing, software
19    distributed under the License is distributed on an "AS IS" BASIS,
20    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
21    See the License for the specific language governing permissions and
22    limitations under the License. -->
23
24 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
25   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
26   xmlns:xs="http://www.w3.org/TR/2008/REC-xml-20081126#">
27
28     <xsl:strip-space elements="*"/>
29     <xsl:output method="xml" indent="yes"/>
30
31     <xsl:param name="BaseURI"><!-- define if not completed by application-->
32         <!-- e.g. http://www.exampleURI.net/STELLA-I -->
33     </xsl:param>
34
35     <!-- Begin RDF document -->
36     <xsl:template match="/">
37         <xsl:element name="rdf:RDF">
38             <rdf:Description>
39                 <xsl:attribute name="rdf:about"/>
40                 <xsl:apply-templates select="/*|@*"/>
41             </rdf:Description>
42         </xsl:element>
43     </xsl:template>
44
45     <!-- Create triples for elements, turn elements into resources -->
46     <xsl:template match="*">
47         <xsl:param name="subjectname"/>
48
49         <!-- Build URI from ancestor elements -->
50         <xsl:variable name="newssubjectname">
51             <xsl:if test="$subjectname=' '">
52                 <xsl:value-of select="$BaseURI"/>
53                 <xsl:text>#</xsl:text>
54             </xsl:if>
55             <xsl:value-of select="$subjectname"/>
56             <xsl:value-of select="name()"/>
57             <!-- Add unique ID for sibling elements with same tag -->
58             <xsl:variable name="number">
59                 <xsl:number/>
60             </xsl:variable>
61             <xsl:if test="$number > 1">
62                 <xsl:text>_</xsl:text>
63                 <xsl:number/>
64             </xsl:if>
65         </xsl:variable>
66
67         <xsl:element name="{name()}" namespace="{concat(namespace-uri(),'#')}">
68             <rdf:Description>

```

```

69             <xsl:attribute name="rdf:about">
70                 <xsl:value-of select="$newsubjectname" />
71             </xsl:attribute>
72             <xsl:apply-templates select="@*|node()">
73                 <xsl:with-param name="subjectname"
74                     select="concat($newsubjectname, '/')"/>
75             </xsl:apply-templates>
76         </rdf:Description>
77     </xsl:element>
78
79     <!-- Create ordering triples with rdf:_no predicates,
80         comment out if not needed -->
81     <xsl:if test="count(../*) >1">
82         <xsl:element name="{concat('rdf:_', count(preceding-sibling::*)+1)}">
83             <rdf:Description>
84                 <xsl:attribute name="rdf:about">
85                     <xsl:value-of select="$newsubjectname" />
86                 </xsl:attribute>
87             </rdf:Description>
88         </xsl:element>
89     </xsl:if>
90 </xsl:template>
91
92     <!-- Create triples for attributes -->
93     <xsl:template match="@*" name="attributes">
94         <xsl:variable name="ns">
95             <!-- If attribute doesn't have a namespace use element namespace -->
96             <xsl:choose>
97                 <xsl:when test="namespace-uri()='' ">
98                     <xsl:value-of select="concat(namespace-uri(..), '#')"/>
99                 </xsl:when>
100                <xsl:otherwise>
101                    <xsl:value-of select="concat(namespace-uri(), '#')"/>
102                </xsl:otherwise>
103            </xsl:choose>
104        </xsl:variable>
105        <xsl:element name="{name()}" namespace="{${ns}}">
106            <xsl:value-of select=". "/>
107        </xsl:element>
108    </xsl:template>
109
110    <!-- Create triples for text with rdf:value predicates -->
111    <xsl:template match="text()">
112        <xsl:element name="rdf:value">
113            <xsl:value-of select=". "/>
114        </xsl:element>
115    </xsl:template>
116
117    <!-- Create triples for comments -->
118    <xsl:template match="comment()">
119        <xsl:element name="xs:comment">
120            <xsl:value-of select=". "/>
121        </xsl:element>
122    </xsl:template>
123
124 </xsl:stylesheet>

```